

Analysis of Financial Data using Non-Negative Matrix Factorization^{*†}

Konstantinos Drakakis, Scott Rickard, Ruairí de Fréin, Andrzej Cichocki

Abstract

We apply Non-negative Matrix Factorization (NMF) to the problem of identifying underlying trends in stock market data. NMF is a recent and very successful tool for data analysis including image and audio processing; we use it here to decompose a mixture of data, the daily closing prices of the 30 stocks which make up the Dow Jones Industrial Average, into its constituent parts, the underlying trends which govern the financial marketplace. We demonstrate how to impose appropriate sparsity and smoothness constraints on the components of the decomposition. Also, we describe how the method clusters stocks together in performance-based groupings which can be used for portfolio diversification.

1 Introduction

The explosion in popularity of Non-negative Matrix Factorization (NMF) in recent times has led to it being applied to diverse fields such as PET [1], EEG analysis [11], pattern recognition, feature extraction, denoising, dimensionality reduction and blind source separation [4].

NMF considers the following problem [8]:

Problem 1 *Given matrix $\mathbf{Y} = [\mathbf{y}(1), \mathbf{y}(2), \dots, \mathbf{y}(T)] \in \mathbb{R}^{m \times T}$, NMF decomposes \mathbf{Y} into the product of two matrices, $\mathbf{A} \in \mathbb{R}^{m \times r}$ and $\mathbf{X} = [\mathbf{x}(1), \mathbf{x}(2), \dots, \mathbf{x}(T)] \in \mathbb{R}^{r \times T}$, where all matrices have exclusively non-negative elements.*

We can view \mathbf{Y} as a mixture of the rows of \mathbf{X} weighted by the coefficients in the mixing matrix \mathbf{A} . Thus, the rows of \mathbf{X} can be thought of as underlying components from which the mixtures are created. We shall consider here decompositions which are approximative in nature, i.e.,

$$\mathbf{Y} = \mathbf{AX} + \mathbf{V}, \quad (1)$$

where $\mathbf{A} \geq 0$, $\mathbf{X} \geq 0$ component-wise and $\mathbf{V} \in \mathbb{R}^{m \times T}$ represents a noise or error matrix.

One arena in which an understanding of the hidden components which drive the system is crucial (and potentially lucrative) is financial markets. It is widely believed that the stock market and indeed individual stock prices are determined by fluctuations in underlying but unknown factors (signals). If one could determine and predict these components, one would have the opportunity to leverage this knowledge for financial gain. In this paper, we use NMF to learn the components which drive the stock market; specifically, we use for \mathbf{Y} the closing prices for the past 20 years of the 30 stocks which make up the Dow Jones Industrial Average. To the best of our knowledge, this is the first time NMF has been applied in a financial setting.

We begin with an overview of the NMF algorithm. Subsequently, we use its 2 standard varieties (Frobenius and Kullback-Leibler) to represent each stock as a weighted sum of underlying components, and propose a novel method for those components to be smooth. In fact, in some instances we are interested in decompositions with sparse \mathbf{A} and smooth \mathbf{X} , and we modify the classical NMF techniques accordingly. Finally, based on the identified mixing matrix \mathbf{A} , we propose a clustering that can be of potential use to investors as it produces a performance-based grouping of the stocks instead of the tradition sector-based grouping.

2 Non-Negative Matrix Factorization

We start by introducing two standard NMF techniques proposed by Lee and Seung [8]. In their seminal work on NMF, [9] considered the squared Frobenius norm and the Kullback-Leibler (KL) objective functions.

^{*}Keywords: Non-negative Matrix Factorization (NMF), Dow-Jones Industrial Average, portfolio diversification, sparsity, smoothness, clustering

[†]Math. subject classification: 91B06, 91B28, 15A23

Frobenius:

$$D_F(\mathbf{Y}||\mathbf{AX}) = \frac{1}{2} \sum_{ik} |y_{ik} - [\mathbf{AX}]_{ik}|^2 \quad (2)$$

Kullback-Leibler:

$$D_{KL}(\mathbf{Y}||\mathbf{AX}) = \sum_{ik} \left(y_{ik} \ln \frac{y_{ik}}{[\mathbf{AX}]_{ik}} + [\mathbf{AX}]_{ik} - y_{ik} \right) \quad (3)$$

A suitable step-size parameter was proposed in both cases resulting in two alternating, multiplicative, gradient descent updating algorithms:

Frobenius:

$$\mathbf{A} \leftarrow \mathbf{A} \odot \mathbf{YX}^T \oslash \mathbf{AXX}^T, \quad (4)$$

$$\mathbf{X} \leftarrow \mathbf{X} \odot \mathbf{A}^T \mathbf{Y} \oslash \mathbf{A}^T \mathbf{AX} \quad (5)$$

Kullback-Leibler:

$$\mathbf{A} \leftarrow \mathbf{A} \odot (\mathbf{Y} \oslash \mathbf{AX}) \mathbf{X}^T \oslash [\mathbf{X}\mathbf{1}_T, \dots, \mathbf{X}\mathbf{1}_T]^T, \quad (6)$$

$$\mathbf{X} \leftarrow \mathbf{X} \odot \mathbf{A}^T (\mathbf{Y} \oslash \mathbf{AX}) \oslash [\mathbf{A}^T \mathbf{1}_m, \dots, \mathbf{A}^T \mathbf{1}_m] \quad (7)$$

where \odot represents element-wise multiplication, \oslash represents element-wise division, and $\mathbf{1}_n$ is a column vector with n ones.

Lee and Seung argue that the advantage of having multiplicative updates is that it provides a simple update rule under which \mathbf{A} and \mathbf{X} never become negative, and so projection into the positive orthant of the space is not necessary. Having alternating updates implies that the optimization is no longer convex (NMF is convex if either \mathbf{A} or \mathbf{X} are updated, while the other is known) and therefore the solution it leads to may be neither unique nor optimal (exact).

Alternatively, NMF factorization can be achieved in many other ways: for example, multiplicative or additive gradient descent [9], projected gradient [10], exponentiated gradient [3], 2nd order Newton [12]; and with different costs such as Kullback-Leibler, Frobenius and Amari-alpha divergence [4]; and with additive or projected sparsity [4, 7] and/or orthogonality constraints [6].

2.1 Sparsity and smoothness

When it is known that the components of \mathbf{A} and \mathbf{X} have certain properties (e.g., sparsity), it is advantageous to manipulate the algorithm to produce components with the desired properties. For example, [7] argues that explicitly incorporating a sparseness constraint improves the found decompositions for face image databases. Alternatively, [2] apply a temporal smoothness constraint which they argue produces meaningful physical and physiological interpretations of clinical EEG recordings. We now give a brief overview of how sparsity and smoothness are implemented in practice. At the end of this section, we propose a novel method of producing a decomposition with a sparse mixing matrix and smooth component matrix.

Consider the following constrained optimization problems with the Frobenius and KL cost [4]:

$$\begin{aligned} D_F^{\alpha_A, \alpha_X}(\mathbf{Y}||\mathbf{AX}) &= D_F(\mathbf{Y}||\mathbf{AX}) + \alpha_A J_A(\mathbf{A}) + \alpha_X J_X(\mathbf{X}), \\ D_{KL}^{\alpha_A, \alpha_X}(\mathbf{Y}||\mathbf{AX}) &= D_{KL}(\mathbf{Y}||\mathbf{AX}) + \alpha_A J_A(\mathbf{A}) + \alpha_X J_X(\mathbf{X}), \\ \text{s. t. } \forall i, j, k: x_{jk} &\geq 0, \quad a_{ij} \geq 0, \end{aligned} \quad (8)$$

where $\alpha_A \geq 0$ and $\alpha_X \geq 0$ are regularization parameters, and functions $J_X(\mathbf{X})$ and $J_A(\mathbf{A})$ are used to enforce a certain application-dependent characteristics of the solution, such as sparsity and/or smoothness.

2.1.1 Regularization functions:

The regularization, terms $J_A(\mathbf{A})$ and $J_X(\mathbf{X})$ can be defined in many ways. Let us assume the following definition for L_p -norm of a given matrix $\mathbf{C} = [c_{mn}] \in \mathbb{R}^{M \times N}$: $L_p(\mathbf{C}) \triangleq \left(\sum_{m=1}^M \sum_{n=1}^N |c_{mn}|^p \right)^{\frac{1}{p}}$, thus:

- **L_1 norm** ($p = 1$) – for sparse solution:

$$\begin{aligned} J_A(\mathbf{A}) &= \sum_{i=1}^m \sum_{j=1}^r a_{ij}, \quad J_X(\mathbf{X}) = \sum_{j=1}^r \sum_{k=1}^T x_{jk}, \\ \nabla_A J_A(\mathbf{A}) &= \mathbf{1}, \quad \nabla_X J_X(\mathbf{X}) = \mathbf{1}, \end{aligned}$$

- **Squared L_2 norm** ($p = 2$) – for smooth solution:

$$J_A(\mathbf{A}) = \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^r a_{ij}^2, \quad J_X(\mathbf{X}) = \frac{1}{2} \sum_{j=1}^r \sum_{k=1}^T x_{jk}^2,$$

$$\nabla_A J_A(\mathbf{A}) = a_{ij}, \quad \nabla_X J_X(\mathbf{X}) = x_{jk},$$

There exists many other measures of sparsity and smoothness (e.g., see in particular [7] which proposes a sparse measure with intuitively desirable properties), each of which has associated modified algorithms.

2.1.2 Modified Frobenius and KL algorithms:

The Lee-Seung algorithm with Frobenius cost can be regularized using additive penalty terms [4]. Using gradient descent the following generalized updates can be derived:

$$a_{ij} \leftarrow a_{ij} \frac{[\mathbf{Y} \mathbf{X}^T]_{ij} - \alpha_A [\nabla_A J_A(\mathbf{A})]_{ij}]_{\varepsilon}}{[\mathbf{A} \mathbf{X} \mathbf{X}^T]_{ij}}, \quad (9)$$

$$x_{jk} \leftarrow x_{jk} \frac{[\mathbf{A}^T \mathbf{Y}]_{jk} - \alpha_X [\nabla_X J_X(\mathbf{X})]_{jk}]_{\varepsilon}}{[\mathbf{A}^T \mathbf{A} \mathbf{X}]_{jk}}, \quad (10)$$

$$a_{ij} \leftarrow \frac{a_{ij}}{\sum_{i=1}^m a_{ij}}, \quad (11)$$

where the nonlinear operator $[x]_{\varepsilon} = \max\{\varepsilon, x\}$ has been included in the numerator to avoid the possibility of a negative numerator. Note that, in general, the signature matrix \mathbf{A} is normalized at the end of each pair (\mathbf{A} and \mathbf{X}) of updates.

Regularization of the Lee and Seung algorithm with Kullback-Leibler cost function was considered in [5]. The modified learning rules were found to perform poorly. In order to enforce sparsity while using the Kullback-Leibler cost function, a different approach can be used [4]:

$$a_{ij} \leftarrow \left(a_{ij} \frac{\sum_{k=1}^T x_{jk} (y_{ik} / [\mathbf{A} \mathbf{X}]_{ik})}{\sum_{p=1}^T x_{jp}} \right)^{1+\alpha_{Sa}}, \quad (12)$$

$$x_{jk} \leftarrow \left(x_{jk} \frac{\sum_{i=1}^m a_{ij} (y_{ik} / [\mathbf{A} \mathbf{X}]_{ik})}{\sum_{q=1}^m a_{qj}} \right)^{1+\alpha_{Sx}}, \quad (13)$$

$$a_{ij} \leftarrow \frac{a_{ij}}{\sum_{i=1}^m a_{ij}}, \quad (14)$$

where α_{Sa} and α_{Sx} enforce sparse solutions. Typically, $\alpha_{Sa}, \alpha_{Sx} \in [0.001, 0.005]$. This is a rather intuitive or heuristic approach to sparsification. Raising the Lee-Seung learning rules to the power of $(1 + \alpha_{Sa})$ or $(1 + \alpha_{Sx})$, and having $\alpha_{Sa}, \alpha_{Sx} > 0$ resulting in an exponents that are greater than one, implies that the small values in the non-negative matrix tend to zero as the number of iterations increase. The components of the non-negative matrix that are greater than one are forced to be larger. In practice these learning rules are stable and the algorithm works well.

Of course, this power raising technique can also be used with the Frobenius norm, if we wish: in both cases then, this reduces to an intermediate step of the algorithm performing the updates:

$$x_{ij} \leftarrow x_{ij}^{1+\alpha_{Sx}}, \quad a_{ij} \leftarrow a_{ij}^{1+\alpha_{Sa}}. \quad (15)$$

This is actually the smoothness and sparsity technique we will be using to derive our results.

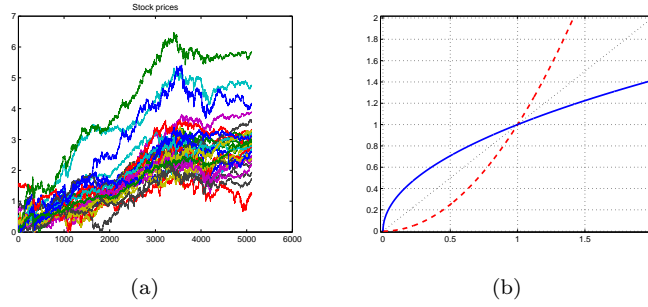


Figure 1: (a) The original stock prices data; (b) graph of the functions $f(x) = x^2$ and $f(x) = \sqrt{x}$, $x > 0$.

2.1.3 Sparse \mathbf{A} and smooth \mathbf{X}

In many practical situation, we may desire a decomposition which has sparse \mathbf{A} and smooth \mathbf{X} (or, alternatively sparse \mathbf{X} and smooth \mathbf{A}). We now propose a simple approach which produces such solutions. In the Frobenius case, we required a projection to correct the modified algorithm's solutions that occasionally lie outside the allowed non-negative space. The modified Kullback-Leibler method does not produce satisfactory results which led to the heuristic and intuitive approach for sparsification of the solution [5]. If we, however, desire smooth solutions, we can modify (12) or (13) simply by setting $\alpha_{Sa} < 0$ or $\alpha_{Sx} < 0$. By setting, for example, $\alpha_{Sa} > 0$ and $\alpha_{Sx} < 0$, the solutions produced have sparse \mathbf{A} and smooth \mathbf{X} . This is the approach we use in this paper.

3 Stock market trends

We will be applying NMF on the closing prices for the past 20 years of the 30 stocks that make up the Dow Jones Industrial Index. These stocks are traditionally grouped into *sectors* according to the nature of the activities or the object of trade of the company they correspond to (such as Technology, Basic Materials, Financial etc.); full information about these stocks can be found in Table 3. The closing prices of stocks are definitely nonnegative signals, so it makes sense to apply NMF on them; we actually analyze their natural logarithms, and, to obtain better results, we “ground” them, namely we shift the time series of each stock vertically so that the minimum value becomes 0.

It is reasonable to assume that stocks that are part of the same market do not behave independently of each other, but are rather driven by some underlying forces, normally significantly fewer than the number of stocks themselves. In this way, groups of stocks exhibit a correlated behavior, and these groups may or may not coincide with the sectors mentioned above. Obviously, the most interesting scenario would be a high performance group almost “orthogonal” to the sectors (that is, containing one stock from most sectors), as investment in such a group would falsely appear to be diversified and subject investors to unsuspected risk.

NMF is definitely an exemplary candidate method to determine these underlying driving components; after all, this is exactly what it has been successfully applied to in audio and image processing, where it has proved extremely efficient in demixing speech and separating overlapping images. In what follows, we will test some of the variants of NMF we have been describing above on our financial data.

One last point to be mentioned concerns the clustering algorithm that determines the groups of the stocks; this is, however, an extensively studied problem in statistics and most mathematical software packages include efficient algorithms that solve it automatically. In our case, we use the *kmeans* routine of MATLAB on the matrix \mathbf{A} we determine at the end of each run.

4 Results

We divide this section into various parts, according to the features incorporated in the version of the NMF algorithm we use. Note that \mathbf{X} is always normalized so that each row's maximum is equal to 1. We run the algorithms through a recursion of 1000 steps, or until the cost stops decreasing, whichever comes first. In Fig. 1 we show 2 things that should be kept in mind throughout the experiments: the original stock prices, so that we can compare the reconstructions we obtain through NMF, and the action of exponentiation, whereby exponents larger than 1 increase the range of values (making large values larger and small values smaller, relatively speaking), while exponents less than 1 suppress it.

#	Symbol	Company	Sector
1	AA	Alcoa	Basic Materials
2	AIG	American International Group	Financial
3	AXP	American Express	Financial
4	BA	Boeing	Industrial Goods
5	C	Citigroup	Financial
6	CAT	Caterpillar	Industrial Goods
7	DD	DuPont	Basic Materials
8	DIS	Disney	Services
9	GE	General Electric	Conglomerates
10	GM	General Motors	Consumer Goods
11	HD	Home Depot	Services
12	HON	Honeywell	Industrial Goods
13	HPQ	Hewlett-Packard	Technology
14	IBM	International Business Machines	Technology
15	INTC	Intel	Technology
16	JNJ	Johnson and Johnson	Healthcare
17	JPM	JP Morgan Chase	Financial
18	KO	Coca-Cola	Consumer Goods
19	MCD	McDonald's	Services
20	MMM	Minnesota Mining and Manufacturing	Conglomerates
21	MO	Altria	Consumer Goods
22	MRK	Merck	Healthcare
23	MSFT	Microsoft	Technology
24	PFE	Pfizer	Healthcare
25	PG	Procter and Gamble	Consumer Goods
26	T	AT&T	Technology
27	UTX	United Technologies	Conglomerates
28	VZ	Verizon	Technology
29	WMT	Wal-Mart	Services
30	XOM	ExxonMobil	Basic Materials

Table 1: The 30 stocks of the Dow Jones Industrial Average (for the past 20 years) and their sectors.

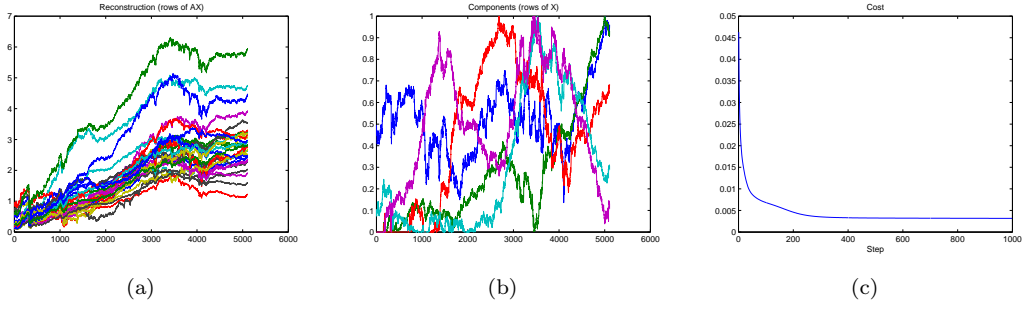


Figure 2: Results of the Frobenius run with no extra smoothness or sparsity

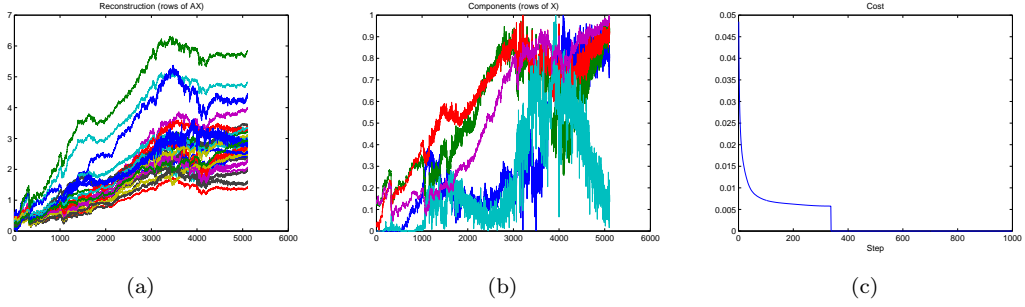


Figure 3: Results of the Frobenius run with no extra smoothness but extra sparsity for \mathbf{A}

4.1 Frobenius runs

In the following runs we use the Frobenius norm:

4.1.1 No smoothness, no sparsity

In this run we use “vanilla” NMF, making no effort to induce either extra sparsity or extra smoothness; in other words, $\alpha_{Sx} = \alpha_{Sa} = 0$ in (15). The recursion goes through all 1000 steps, and the results are shown in Fig.2. We see that the components (rows of \mathbf{X}) are already sufficiently smooth, and that the reconstruction of the data from the determined components is also very good. The cost is computed using (2).

4.1.2 No smoothness, sparsity

In this run we use $\alpha_{Sx} = 0$, $\alpha_{Sa} = 0.01$ in (15) in order to obtain a sparse \mathbf{A} ; this turns out to produce components which oscillate too much (see Fig.3). Hence, extra sparsity for \mathbf{A} should be used in parallel with extra smoothness for \mathbf{X} . Note also that the recursion ended prematurely here, indicating that the cost reached a minimum and was about to increase again.

4.1.3 Smoothness, no sparsity

In this run we use $\alpha_{Sx} = -0.05$, $\alpha_{Sa} = 0$ in (15) in order to obtain a smooth \mathbf{X} ; this time the recursion goes on till the end, the components are smooth, but they are also very similar to each other, with a clear upward trend (see Fig.4). If we remove this common upward trend, the components will look similar to those in Fig.2.

4.1.4 Other attempts to induce sparsity and smoothness

We saw earlier that our attempt to sparsify \mathbf{A} by exponentiation led to oscillatory components in \mathbf{X} . We can remove this oscillation either by using $\alpha_{Sx} < 0$, or by “brute force”, namely by convolving the rows of \mathbf{X} with a smoothing window at the end of each step of the algorithm. The latter method is especially appealing, as we can smoothen the components to an arbitrary degree, by choosing the width of our smoothening window (we used a Hamming window of width 101). Here, the components look very much like the ones in Fig.4.

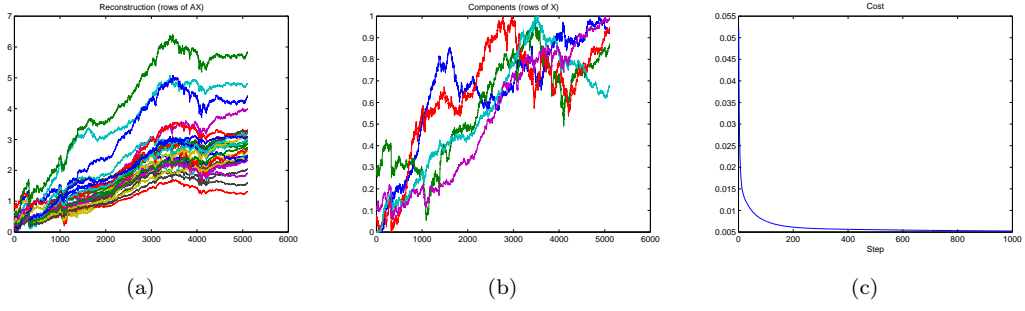


Figure 4: Results of the Frobenius run with no extra sparsity but extra smoothness for \mathbf{X}

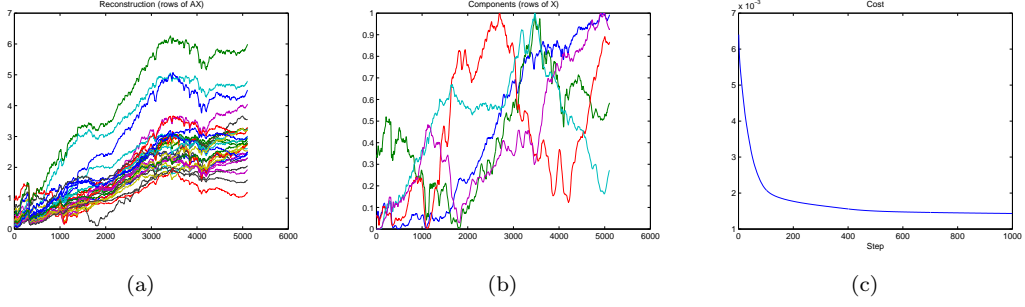


Figure 5: Results of the Kullback-Leibler run with no extra smoothness or sparsity

4.2 Kullback-Leibler runs

In the following runs we use the Kullback-Leibler norm:

4.2.1 No smoothness, no sparsity

In this run we use again “vanilla” NMF, making no effort to induce either extra sparsity or extra smoothness: we set $\alpha_{Sx} = \alpha_{Sa} = 0$ in (15). The recursion goes through all 1000 steps, and the results are shown in Fig.5. We see that the components (rows of \mathbf{X}) are already sufficiently smooth, and that the reconstruction of the data from the determined components is also very good. The cost is computed using (3).

4.2.2 No smoothness, sparsity

In this run we use $\alpha_{Sx} = 0$, $\alpha_{Sa} = 0.002$ in (15) in order to obtain a sparse \mathbf{A} ; given the high oscillation we observed in the corresponding Frobenius run, we decreased the exponent used, and this time the result is satisfactory. Although the components are more oscillatory than before, they do not oscillate wildly (see Fig.6). Finally, note that the recursion ended prematurely here.

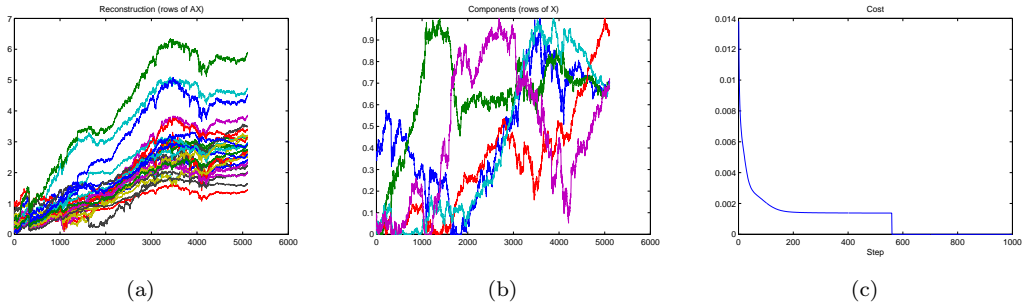


Figure 6: Results of the Kullback-Leibler run with no extra smoothness but extra sparsity for \mathbf{A}

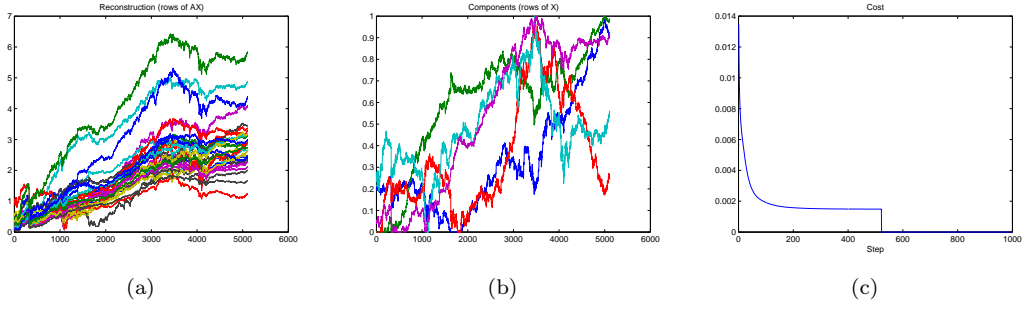


Figure 7: Results of the Kullback-Leibler run with extra smoothness for \mathbf{X} and extra sparsity for \mathbf{A}

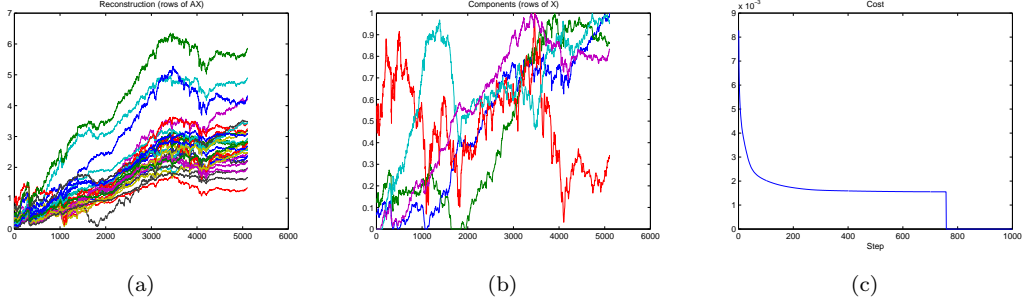


Figure 8: Results of the Kullback-Leibler run with extra smoothness induced by filtering for \mathbf{X} and extra sparsity for \mathbf{A}

4.2.3 Smoothness and sparsity

In this run we use $\alpha_{Sx} = -0.002$, $\alpha_{Sa} = 0.002$ in (15) in order to obtain a sparse \mathbf{A} ; the results are very nice, as Fig.7 shows.

4.2.4 Smoothness by convolution and sparsity

In this run we use $\alpha_{Sx} = 0$, $\alpha_{Sa} = 0.003$ in (15), but attempt to smoothen the components by filtering. As the convolution with a large Hamming window did not produce interesting results in the corresponding Frobenius runs, we now use a very small Hamming window instead of width 5 (after all, we are only interested in fine scale smoothing).

4.3 Discussion of the various runs

In the experiments above we tried to sample a huge variety of parameters. Our results suggest the following:

- There is practically no difference between using the Frobenius cost and using the Kullback-Leibler cost;
- Attempts to sparsify \mathbf{A} lead to oscillations in \mathbf{X} that need to be smoothened out;
- Smoothening through exponentiation tends to perform coarse scale smoothening as well as fine scale smoothening and reduces the overall oscillation of the components, yielding very similar components which need to have their trends removed;
- Smoothening \mathbf{X} through filtering (with a Hamming window, in our example) gives us more control which scales we want to smoothen over, and is thus a method more suitable for our purposes.

4.4 Stock clustering

The determination of the underlying driving components of the market in the various runs above leads us to investigate which stocks behave similarly, in the sense that they rely mainly on (are linear combinations of, that is) the same components. This can be easily done by examining the rows of \mathbf{A} determined at the end of each run and deciding which rows have the dominant coefficients at the same positions (columns). This is a

3, 5	11, 23	13, 14, 17	2, 9, 15, 24	16, 21, 25, 29	1, 6, 20, 27, 30	4, 7, 8, 10, 12, 18, 19, 22, 26, 28
15	11, 23	4, 25, 30	6, 16, 20	8, 18, 21, 22, 29	2, 3, 5, 9, 14, 24, 27	1, 7, 10, 12, 13, 17, 19, 26, 28
13, 17	4, 8, 18	3, 5, 6, 27	11, 15, 22, 23	16, 20, 21, 25, 30	1, 2, 9, 14, 24, 29	7, 10, 12, 19, 26, 28
15	14, 17	11, 23	6, 12, 13, 27	2, 3, 5, 9, 24	16, 18, 21, 22, 25, 29	1, 4, 7, 8, 10, 19, 20, 26, 28, 30
14	13, 17	3, 6, 27	11, 15, 23	2, 5, 9, 24, 29	1, 4, 16, 20, 21, 25, 30	7, 8, 10, 12, 18, 19, 22, 26, 28
13, 17	11, 23	3, 5	6, 14, 27	2, 9, 12, 15, 24	7, 8, 10, 19, 26, 28	1, 4, 16, 18, 20, 21, 22, 25, 29, 30
14	11, 15, 23	4, 21, 25	1, 2, 9, 24, 29	3, 5, 6, 13, 17	12, 16, 20, 27, 30	7, 8, 10, 18, 19, 22, 26, 28

Table 2: Clustering of the stocks according to the 7 runs performed, in the same order from top to bottom; the correspondence between numbers and stocks can be found in Table 3.

well studied problem in statistics and MATLAB has a routine (*kmeans*) that does the grouping automatically upon feeding it with \mathbf{A} and the number of clusters we seek; we set 7 clusters in all cases.

The results of the clustering can be seen in Table 4.4, which demonstrates clearly the success of NMF in the analysis of financial data. Indeed, we see that clusters (or at least sub-clusters) of stocks persist across the various runs, that is they are not dependent on the parameters we run NMF with (cost, sparsity, smoothness etc.) This implies that NMF is robust when financial data is concerned, hence very suitable for analysis. In addition, those persistent clusters, as we are about to see, are frequently “orthogonal” to the traditionally defined sectors, falsely suggesting good portfolio diversification options. More precisely, to give some concrete examples,

- 3 and 5 are always grouped together, with 1 exception only (run 5), and correspond to American Express and Citigroup (finance).
- 11 and 23 are always together, and correspond to Microsoft (technology) and Home Depot (services).
- 2, 9, and 24 are always grouped together, often along with 29, and correspond to AIG (financial), General Electric (conglomerates), Pfizer (healthcare), and Wal-Mart (services), so the group they form cuts across 4 sectors.
- 21 and 25 are always together with 1 exception (run 2), and correspond to Altria and Procter & Gamble, both in consumer goods.
- 13 and 17 are always together except in run 4, and correspond to Hewlett-Packard (technology) and JP Morgan Chase (finance).
- 7, 10, 19, 26, 28 are grouped together in all runs, often along with 12 and 22: this is a large group comprising DuPont (basic materials), General Motors (consumer goods), MacDonald’s (services), AT&T (technology), Verizon (technology), Honeywell (industrial goods), and Merck (healthcare), and truly remarkable as it intersects almost all sectors.

5 Conclusion and summary

We tested the main variations of the NMF algorithm on the analysis of financial data, namely the closing prices of the 30 stocks forming the Dow Jones Industrial Index for the past 20 years, in order to discover the underlying forces driving the financial market. We found that there is a tradeoff between the smoothness of those underlying components and the sparsity of the representation of the stock prices as linear combinations of these components, and proposed a novel method for sparsity and smoothness through exponentiation.

But, more importantly, we also found that the clustering of the stocks according to the components they depend on is robust in the parameters we use, in the sense that certain groups of stocks persist in all (or at least in most) clusterings we attempted, and that those groups often span many sectors, thus potentially giving investors a false sense of security that they have diversified their portfolio, when in fact they have not. Conversely, investing in stocks belonging to different persistent groups, but not necessarily in different sectors, offers genuine portfolio diversification options.

References

- [1] B. Bodvarsson and M. Mørkebjerg. Analysis of dynamic PET data. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2006. Supervised by Lars Kai Hansen, IMM.
- [2] Z. Chen, A. Cichocki, and T. Rutkowski. Constrained non-negative matrix factorization method for EEG analysis in early detection of alzheimer disease. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 5, pages 893–896, May 2006.
- [3] A. Cichocki, S. Amari, R. Zdunek, R. Kompass, G. Hori, and Z. He. Extended SMART algorithms for non-negative matrix factorization. In *ICAISC*, pages 548–562, 2006.
- [4] A. Cichocki and R. Zdunek. *NMFLAB User's Guide MATLAB Toolbox for Non-Negative Matrix Factorization*. Laboratory for Advanced Brain Signal Processing, Riken, 2006.
- [5] A. Cichocki, R. Zdunek, and S. Amari. New algorithms for non-negative matrix factorization in applications to blind source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 5, pages 621–624, May 2006.
- [6] T. Feng, S. Li, H. Shum, and H. Zhang. Local non-negative matrix factorization as a visual representation. In *ICDL '02: Proceedings of the 2nd International Conference on Development and Learning*, page 178, Washington, DC, USA, 2002. IEEE Computer Society.
- [7] P. Hoyer. Non-negative matrix factorization with sparseness constraints. *J. Mach. Learn. Res.*, 5:1457–1469, 2004.
- [8] D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.
- [9] D. Lee and H. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.
- [10] C. Lin. Projected gradient methods for non-negative matrix factorization. Technical report, Department of Computer Science, National Taiwan University, 2005.
- [11] W. Liu, N. Zheng, and X. Li. *Nonnegative Matrix Factorization for EEG Signal Classification*, volume 3174 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004.
- [12] R. Zdunek and A. Cichocki. Non-negative matrix factorization with quasi-newton optimization. In *ICAISC*, pages 870–879, 2006.

Author information:

Konstantinos Drakakis, Scott Rickard, Ruairí de Fréin
Electronic & Electrical Engineering
University College Dublin
Belfield
Dublin 4
Ireland
Email: {Konstantinos.Drakakis, Scott.Rickard}@ucd.ie, Ruairi.deFrein@ee.ucd.ie

Andrzej Cichocki
Laboratory for Advanced Brain Signal Processing
Brain Science Institute
RIKEN
2-1 Hirosawa, Wako-shi
Saitama, 351-0198
Japan
Email: cia@brain.riken.jp